RIACS

Research Institute for Advanced Computer Science
NASA Ames Research Center

IN-61
43111
p 34

An Implementation of the Look-Ahead Lanczos Algorithm for Non-Hermitian Matrices Part I

Roland W. Freund, Martin H. Gutknecht,
and Noël M. Nachtigal

RIACS Technical Report 90.45

November 1990

An Implementation of the Look-Ahead Lanczos Algorithm for Non-Hermitian Matrices Part I

Roland W. Freund, Martin H. Gutknecht,

and Noël M. Nachtigal

The Research Institute for Advanced Computer Science is operated by Universities Space Research Association (USRA), The American City Building, Suite 311, Columbia, MD 21044, (301)730-2656.

Work reported herein was supported in part by DARPA via Cooperative Agreement NCC 2-387 between NASA and USRA.

An Implementation of the Look-Ahead Lanczos Algorithm for Non-Hermitian Matrices Part I

ROLAND W. FREUND
Research Institute for Advanced Computer Science

MARTIN H. GUTKNECHT
Interdisciplinary Project Center for Supercomputing

NOËL M. NACHTIGAL
Massachusetts Institute of Technology

The nonsymmetric Lanczos method can be used to compute eigenvalues of large sparse non-Hermitian matrices or to solve large sparse non-Hermitian linear systems. However, the original Lanczos algorithm is susceptible to possible breakdowns and potential instabilities. We present an implementation of a look-ahead version of the Lanczos algorithm which overcomes these problems by skipping over those steps in which a breakdown or near-breakdown would occur in the standard process. The proposed algorithm can handle look-ahead steps of any length and is not restricted to steps of length 2, as earlier implementations are. Also, our implementation has the feature that it requires roughly the same number of inner products as the standard Lanczos process without look-ahead.

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra–*eigenvalues; linear systems (direct and iterative methods); sparse and very large systems;* G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Large sparse linear systems and eigenvalue problems, iterative methods

Additional Key Words and Phrases: Non-Hermitian matrices, Lanczos method, look-ahead steps

The work of R. W. Freund and N. M. Nachtigal was supported in part by DARPA via Cooperative Agreement NCC 2-387 between NASA and the Universities Space Research Association (USRA).

Authors' addresses: R. W. Freund, RIACS, Mail Stop Ellis Street, NASA Ames Research Center, Moffett Field, CA 94035, and Institut für Angewandte Mathematik, Universität Würzburg, D–W8700 Würzburg, Federal Republic of Germany; M. H. Gutknecht, Interdisciplinary Project Center for Supercomputing, ETH Zürich, ETH-Zentrum, CH-8092 Zürich, Switzerland; N. M. Nachtigal, Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139.

1. INTRODUCTION

In 1950, Lanczos [14] proposed a method for successive reduction of a given, in general non-Hermitian, $N \times N$ matrix A to tridiagonal form. More precisely, the Lanczos procedure generates a sequence $H^{(n)}$, $n = 1, 2, \ldots$, of tridiagonal $n \times n$ matrices which, in a certain sense, approximate A. Furthermore, in exact arithmetic and if no breakdown occurs, the Lanczos method terminates after at most n ($\leq N$) steps with $H^{(n)}$ a tridiagonal matrix which represents the restriction of A or A^T to an A-invariant or A^T-invariant subspace of $\mathbf{C}^N$, respectively. In particular, all eigenvalues of $H^{(n)}$ are also eigenvalues of A, and, in addition, the method also produces basis vectors for the A-invariant or A^T-invariant subspace found.

In the Lanczos process, the matrix A itself is never modified and it appears only in the form of matrix-vector products $A \cdot v$ and $A^T \cdot w$. Because of this feature, the method is especially attractive for sparse matrix computations. Indeed, in practice, the Lanczos process is mostly applied to large sparse matrices A, either for computing eigenvalues of A or — in the form of the closely related biconjugate gradient (BCG) algorithm [15] — for solving linear systems $Ax = b$. For large A, the finite termination property is of no practical importance and the Lanczos method is used as a purely iterative procedure. Typically, the spectrum of $H^{(n)}$ offers good approximations to some of the eigenvalues of A after already relatively few iterations, *i.e.* for $n \ll N$. Similarly, BCG — especially if used in conjunction with preconditioning — often converges in relatively few iterations to the solution of $Ax = b$.

Unfortunately, in the standard Lanczos method a breakdown — more precisely, division by 0 — may occur before an invariant subspace is found. In finite precision arithmetic, such exact breakdowns are very unlikely; however, near-breakdowns may occur which lead to numerical instabilities in subsequent iterations. The possibility of breakdowns has brought the nonsymmetric Lanczos process into discredit and has certainly prevented many people from using the algorithm on non-Hermitian matrices. Note that the symmetric Lanczos process for Hermitian matrices A is a special case of the general procedure in which the occurrence of breakdowns can be excluded.

On the other hand, it is possible to modify the Lanczos process such that it skips over those iterations in which exact breakdown would occur in the standard method. This was already observed by Gragg [8, pp. 222–223] and, in the context of the partial realization problem, by Kung [13, Chapter IV] and Gragg and Lindquist [9]. However, a complete treatment of the modified Lanczos method and its intimate connection with orthogonal polynomials and Padé approximation was presented only recently, by Gutknecht [10, 11]. Clearly, in finite-precision arithmetic, a viable modified Lanczos process also needs to skip over near-breakdowns. Taylor [19] and Parlett, Taylor, and Liu [18], with their look-ahead Lanczos algorithm, were the first to propose such a practical procedure. However, in [19, 18], the details of an actual implementation are worked out only for look-ahead steps of length 2. We will use the term *look-ahead Lanczos method* in a broader sense to denote extensions of the standard Lanczos process which skip over breakdowns. Finally, note that, in addition to [10], there are several other recent papers dealing with various aspects of look-ahead Lanczos methods (see [1, 2, 5, 7, 12, 17]).

The main purpose of this paper is to present a robust implementation, including FORTRAN code, of the look-ahead Lanczos method for general complex non-Hermitian matrices. Our intention was to develop an algorithm which can be used as a black box. In particular, the code can handle look-ahead steps of any length and is not restricted to steps of length 2. On many modern computer architectures, the computation of inner products of long vectors is a bottleneck. Therefore, one of our objectives was to minimize the number of inner products in our implementation of the look-ahead Lanczos method. Indeed, the proposed algorithm requires the same number of inner products as the classical Lanczos process, as opposed to the look-ahead algorithm described in [19, 18], which always requires additional inner products. In particular, our implementation differs from the one in [19, 18] even for look-ahead steps of length 2.

This paper consists of Part I and Part II. The outline of the Part I is as follows. In Section 2, we recall the standard nonsymmetric Lanczos method and its close relationship with orthogonal polynomials. Using this connection, we then describe the basic idea of the look-ahead versions of the Lanczos process. In Section 3, some further notation is introduced. In Section 4, we outline the sequential look-ahead algorithm, and in Section 5, we give details of its actual implementation. In Section 6, we sketch the block version of the look-ahead Lanczos method. In Section 7, we make some concluding remarks.

In Part II [6] of the paper, we describe how the look-ahead Lanczos process can be used to compute approximate solutions to $Ax = b$, solutions which are defined by a quasi-minimal residual (QMR) property. We also show that BCG iterates — when they exist — can be easily obtained from quantities generated by the QMR method. Moreover in Part II, we report numerical experiments with the sequential look-ahead Lanczos algorithm applied to nonsymmetric eigenvalue problems and to solving nonsymmetric linear systems. Finally, the actual codes for the sequential look-ahead algorithm and for the associated QMR algorithm appear in the Appendix to Part II.

Throughout the paper, all vectors and matrices, unless otherwise stated, are assumed to be complex. As usual, $M^T = (m_{ji})$ and $M^H = (\overline{m}_{ji})$ denote the transpose and the conjugate transpose, respectively, of the matrix $M = (m_{ij})$. The set of singular values of M is denoted by $\sigma(M)$, with $\sigma_{\max}(M)$ and $\sigma_{\min}(M)$ the largest and smallest singular value of M, respectively. The vector norm $\|x\| = \sqrt{x^H x}$ is always the Euclidean norm and $\|M\| = \sigma_{\max}(M)$ denotes the corresponding matrix norm. Moreover, the notation

$$K_n(c, B) := \operatorname{span}\{c, Bc, \ldots, B^{n-1}c\}$$

is used for the nth Krylov subspace of $\mathbf{C}^N$ generated by $c \in \mathbf{C}^N$ and the $N \times N$ matrix B.

$$\mathcal{P}_n := \{\Psi(\lambda) \equiv \gamma_0 + \gamma_1\lambda + \cdots + \gamma_n\lambda^n \mid \gamma_0, \gamma_1, \ldots, \gamma_n \in \mathbf{C}\}$$

denotes the set of all complex polynomials of degree at most n. Finally, it is always assumed that A is a complex, in general non-Hermitian, $N \times N$ matrix.

2. BACKGROUND

In this section, we briefly recall the classical nonsymmetric Lanczos method [14] and its close relationship with *formally orthogonal polynomials* (FOPs hereafter). Using this connection, we then describe the basic idea of the look-ahead Lanczos algorithm.

Given two non-zero starting vectors $v_1 \in \mathbf{C}^N$ and $w_1 \in \mathbf{C}^N$, the standard nonsymmetric Lanczos method generates two sequences of vectors $\{v_n\}_{n=1}^L$ and $\{w_n\}_{n=1}^L$ such that, for $n = 1, \ldots, L$,

$$\begin{aligned} \operatorname{span}\{v_1, v_2, \ldots, v_n\} &= K_n(v_1, A), \\ \operatorname{span}\{w_1, w_2, \ldots, w_n\} &= K_n(w_1, A^T), \end{aligned} \tag{2.1}$$

and

$$w_i^T v_j = d_i \delta_{ij}, \quad \text{with } d_i \neq 0, \quad \text{for all} \quad i, j = 1, \ldots, n. \tag{2.2}$$

Here δ_{ij} denotes the Kronecker delta. The actual construction of the vectors v_n and w_n is based on the three-term recurrences

$$\begin{aligned} v_{n+1} &= A v_n - \alpha_n v_n - \beta_n v_{n-1}, \\ w_{n+1} &= A^T w_n - \alpha_n w_n - \beta_n w_{n-1}, \end{aligned} \tag{2.3}$$

where

$$\alpha_n = \frac{w_n^T A v_n}{d_n}, \quad \beta_n = \frac{d_n}{d_{n-1}}, \quad d_n = w_n^T v_n,$$

are chosen to enforce (2.2). Note that, for $n = 1$, we set $\beta_1 = 0$ and $v_0 = w_0 = 0$ in (2.3). Also, letting

$$V^{(n)} = [\, v_1 \quad v_2 \quad \cdots \quad v_n \,] \quad \text{and} \quad W^{(n)} = [\, w_1 \quad w_2 \quad \cdots \quad w_n \,] \tag{2.4}$$

denote the matrix whose columns are the first n of the vectors v_j and w_j, respectively, and letting

$$H^{(n)} := \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ 1 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_n \\ 0 & \cdots & 0 & 1 & \alpha_n \end{bmatrix}$$

denote the tridiagonal matrix containing the recurrence coefficients, we can rewrite (2.3) as

$$\begin{aligned} A V^{(n)} &= V^{(n)} H^{(n)} + [\, 0 \quad \cdots \quad 0 \quad v_{n+1} \,], \\ A^T W^{(n)} &= W^{(n)} H^{(n)} + [\, 0 \quad \cdots \quad 0 \quad w_{n+1} \,]. \end{aligned} \tag{2.5}$$

Moreover, the biorthogonality condition (2.2) reads

$$(W^{(n)})^T V^{(n)} = D^{(n)} = \operatorname{diag}(d_1, d_2, \ldots, d_n). \tag{2.6}$$

Now, let L be the largest integer such that there exist vectors v_n and w_n, $n = 1, \ldots, L$, satisfying (2.1) and (2.2). Note that $L \leq N$ and that, in view of (2.3), L is the smallest integer such that

$$w_{L+1}^T v_{L+1} = 0. \tag{2.7}$$

Moreover, let

$$L_r = L_r(v_1, A) := \dim K_N(v_1, A) \quad \text{and} \quad L_l = L_l(w_1, A^T) := \dim K_N(w_1, A^T) \tag{2.8}$$

denote the *grade* of v_1 with respect to A and the *grade* of w_1 with respect to A^T, respectively (cf. [20, p. 37]). There are two essentially different cases for fulfilling the termination condition (2.7). The first case, referred to as ***regular termination***, occurs when $v_{L+1} = 0$ or $w_{L+1} = 0$. Clearly, if $v_{L+1} = 0$, then $L = L_r$ and the *right* Lanczos vectors $v_1, \ldots, v_{L_r}$ span the A-invariant subspace $K_{L_r}(v_1, A)$. Similarly, if $w_{L+1} = 0$, then the *left* Lanczos vectors $w_1, \ldots, w_{L_l}$ span the A^T-invariant subspace $K_{L_l}(w_1, A^T)$. Unfortunately, it can also happen that the termination condition (2.7) is satisfied with $v_{L+1} \neq 0$ and $w_{L+1} \neq 0$. This second case is referred to as ***serious breakdown*** [20, p. 389]. Note that, in this case,

$$L < L_* := \min\{L_l, L_r\}$$

and the Lanczos vectors span neither an A-invariant nor an A^T-invariant subspace of $\mathbb{C}^N$.

It is the possibility of serious breakdowns, or, in finite precision arithmetic, of ***near-breakdowns***, that has brought the classical nonsymmetric Lanczos algorithm into discredit. However, by means of a look-ahead procedure, it is possible to leap — except for the very special case of an incurable breakdown [19] — over those iterations in which the standard algorithm would break down. Next, using the intimate connection between the Lanczos process and FOPs, we describe the basic idea of the look-ahead Lanczos algorithm.

First, note that

$$\begin{aligned} K_n(v_1, A) &= \{\Psi(A)v_1 \mid \Psi \in \mathcal{P}_{n-1}\}, \\ K_n(w_1, A^T) &= \{\Psi(A^T)w_1 \mid \Psi \in \mathcal{P}_{n-1}\}. \end{aligned} \tag{2.9}$$

In particular, in view of (2.3), for $n = 1, \ldots, L$,

$$v_n = \Psi_{n-1}(A)v_1 \quad \text{and} \quad w_n = \Psi_{n-1}(A^T)w_1, \tag{2.10}$$

where $\Psi_{n-1} \in \mathcal{P}_{n-1}$ is a uniquely defined monic polynomial. Then, introducing the inner product

$$(\Phi, \Psi) := \left(\Phi(A^T)w_1\right)^T \left(\Psi(A)v_1\right) = w_1^T \Phi(A)\Psi(A)v_1 \tag{2.11}$$

and using (2.1), (2.9), and (2.10), we can rewrite the biorthogonality condition (2.2) in terms of polynomials:

$$(\Psi_{n-1}, \Psi) = 0 \quad \text{for all} \quad \Psi \in \mathcal{P}_{n-2} \tag{2.12}$$

and

$$(\Psi_{n-1}, \Psi_{n-1}) \neq 0. \tag{2.13}$$

Note that, except for the Hermitian case, *i.e.* $A = A^H$ and $w_1 = \overline{v}_1$, the inner product (2.11) is indefinite. Therefore, in the general case, there exist polynomials $\Psi \neq 0$ with "length" $(\Psi, \Psi) = 0$ or even $(\Psi, \Psi) < 0$.

A polynomial $\Psi_{n-1} \in \mathcal{P}_{n-1}$, $\Psi_{n-1} \neq 0$, that fulfills (2.12) is called a FOP (with respect to the inner product (2.11)) of degree $n-1$ (see e.g. [3], [4], [10]). Note that the condition (2.12) is empty for $n = 1$, and hence any $\Psi_0 = \gamma_0 \neq 0$ is a FOP of degree 0. From (2.12),

$$\Psi_{n-1}(\lambda) \equiv \gamma_0 + \gamma_1\lambda + \cdots + \gamma_{n-1}\lambda^{n-1}$$

is a FOP of degree $n-1$ if, and only if, its coefficients $\gamma_0, \ldots, \gamma_{n-1}$ are a nontrivial solution of the linear system

$$\begin{bmatrix} m_0 & m_1 & m_2 & \cdots & m_{n-2} \\ m_1 & \iddots & & \iddots & \vdots \\ m_2 & & \iddots & & \vdots \\ \vdots & \iddots & & & m_{2n-5} \\ m_{n-2} & \cdots & \cdots & m_{2n-5} & m_{2n-4} \end{bmatrix} \begin{bmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{n-2} \end{bmatrix} = -\gamma_{n-1} \begin{bmatrix} m_{n-1} \\ m_n \\ m_{n+1} \\ \vdots \\ m_{2n-3} \end{bmatrix}. \tag{2.14}$$

Here

$$m_j := w_1^T A^j v_1 = (1, \lambda^j), \quad j = 0, 1, \ldots,$$

are the moments associated with (2.11). A FOP Ψ_{n-1} is called ***regular*** if it is uniquely determined by (2.12) up to a scalar, and it is said to be ***singular*** otherwise. Remark that FOPs of degree 0 are always regular. By means of (2.14), one easily verifies that a regular FOP Ψ_{n-1} has maximal degree $n-1$. In particular, a regular FOP is unique if it is required to be monic. Moreover, singular FOPs occur if, and only if, the corresponding linear system (2.14) has a singular coefficient matrix, but is consistent. If (2.14) is inconsistent, then no FOP Ψ_{n-1} exists. This case is referred to as ***deficient***. By relaxing (2.12) slightly, one can define so-called ***deficient FOPs*** (see [10] for details). Simple examples (see Section 13) show that the singular and deficient cases do indeed occur. Thus, regular FOPs Ψ_{n-1} need not exist for every degree $n-1$. We would like to stress that this phenomenon is due to the indefiniteness of (2.11). For a positive definite inner product $(\cdot, \cdot)$, unique monic formally orthogonal polynomials always exist.

Finally, given a regular FOP Ψ_{n-1}, it is easily checked whether a regular FOP of degree n exists. Indeed, using (2.14), one readily obtains the following

Lemma. *Let Ψ_{n-1} be a regular FOP (with respect to the inner product (2.11)) of degree $n-1$. Then, a regular FOP of degree n exists if, and only if, (2.13) is satisfied.*

Let us return to the standard nonsymmetric Lanczos process (2.3). Using (2.7), (2.10), (2.11), and the above lemma, we conclude that the termination index L is the smallest integer L for which there exists no regular FOP of degree L. In particular, a serious breakdown occurs if, and only if, no regular FOP exists for some $L < L_*$.

On the other hand, there is a maximal subset

$$\{n_1, n_2, \ldots, n_J\} \subseteq \{1, 2, \ldots, L_*\}, \quad n_1 := 1 < n_2 < \cdots < n_J \leq L_*, \tag{2.15}$$

such that, for each $j = 1, 2, \ldots, J$, there exists a monic regular FOP $\Psi_{n_j-1} \in \mathcal{P}_{n_j-1}$. Note that $n_1 = 1$ in (2.15) since $\Psi_0(\lambda) \equiv 1$ is a monic regular FOP of degree 0. It is well-known that three successive regular FOPs $\Psi_{n_{j-1}-1}$, Ψ_{n_j-1}, and $\Psi_{n_{j+1}-1}$ are connected via a three-term recurrence. Consequently, setting, in analogy to (2.10),

$$v_{n_j} = \Psi_{n_j-1}(A)v_1 \quad \text{and} \quad w_{n_j} = \Psi_{n_j-1}(A^T)w_1, \tag{2.16}$$

we obtain two sequences of vectors $\{v_{n_j}\}_{j=1}^J$ and $\{w_{n_j}\}_{j=1}^J$ which can be computed by means of three-term recurrences. These vectors will be called *regular* vectors, since they correspond to regular FOPs; the starting vectors v_1 and w_1 are always regular. The look-ahead Lanczos procedure is an extension of the classical nonsymmetric Lanczos algorithm; in exact arithmetic, it generates the vectors v_{n_j} and w_{n_j}, $j = 1, \ldots, J$. If $n_J = L_*$ in (2.15), then these vectors can be complemented to a basis for an A-invariant or A^T-invariant subspace of $\mathbf{C}^N$. An incurable breakdown occurs if, and only if, $n_J < L_*$ in (2.15). Finally, note that

$$\begin{aligned} w_{n_j}^T v = w^T v_{n_j} = 0 \quad \text{for all} \quad v \in K_{n_j-1}(v_1, A),\ w \in K_{n_j-1}(w_1, A^T), \\ j = 1, \ldots, J. \end{aligned} \tag{2.17}$$

The look-ahead procedure we have sketched so far only skips over exact breakdowns. It yields what is called the *nongeneric* Lanczos algorithm in [10]. Of course, in finite precision arithmetic, the look-ahead Lanczos algorithm also needs to leap over near-breakdowns. Roughly speaking, a robust implementation should attempt to generate only the "well-defined" regular vectors. In practice, then, one aims at generating two sequences of vectors $\{v_{n_{j_k}}\}_{k=1}^K$ and $\{w_{n_{j_k}}\}_{k=1}^K$ where

$$\{n_{j_k}\}_{k=1}^K \subseteq \{n_j\}_{j=1}^J, \quad j_1 := 1, \tag{2.18}$$

is a suitable subset of (2.15). Note that, in (2.18), we set $j_1 = 1$, since v_1 and w_1 are always regular. The problem of how to determine the set (2.18) of indices of the "well-defined" regular vectors will be addressed in detail in Section 4.

In order to obtain complete bases for the subspaces $K_n(v_1, A)$ and $K_n(w_1, A^T)$, we need to add vectors

$$v_n \in K_n(v_1, A) \setminus K_{n-1}(v_1, A) \quad \text{and} \quad w_n \in K_n(w_1, A^T) \setminus K_{n-1}(w_1, A^T), \\ n = n_{j_{k-1}} + 1, \dots, n_{j_k} - 1, \; k = 2, 3, \dots, K, \tag{2.19}$$

to the two sequences $\{v_{n_{j_k}}\}_{k=1}^K$ and $\{w_{n_{j_k}}\}_{k=1}^K$, respectively. Clearly, (2.19) guarantees that (2.1) remains valid for the look-ahead Lanczos algorithm. The vectors in (2.19) are called *inner* vectors. Moreover, for each k, the vectors v_n, $n = n_{j_k}, n_{j_k} + 1, \dots, n_{j_{k+1}} - 1$, and correspondingly for w_n, are referred to as the kth *block*. The inner vectors of a block built because of an exact breakdown correspond to singular or deficient FOPs, while the inner vectors of a block built because of a near-breakdown correspond to polynomials which in general are combinations of regular, singular, and deficient FOPs. We will refer to both the regular and the inner vectors v_n and w_n generated by the look-ahead variant as right and left Lanczos vectors, in analogy to the notation of the standard nonsymmetric Lanczos algorithm.

So far, we have not specified how to actually construct the inner vectors. The point is that the inner vectors can be chosen such that the v_n's and w_n's from blocks corresponding to different indices k are still biorthogonal to each other. More precisely, with $V^{(n)}$ and $W^{(n)}$ defined as in (2.4), we have, in analogy to (2.6),

$$(W^{(n)})^T V^{(n)} = D^{(n)}, \; n = n_{j_k} - 1, \; k = 2, 3, \dots, K. \tag{2.20}$$

Here, $D^{(n)}$ is now a nonsingular block diagonal matrix with $k-1$ blocks of respective size $(n_{j_{l+1}} - n_{j_l}) \times (n_{j_{l+1}} - n_{j_l})$, $l = 1, \dots, k-1$. Similarly, (2.5) holds, for $n = n_{j_k} - 1$, $k = 2, 3, \dots, K$, with $H^{(n)}$ (cf. (3.5)) now a block tridiagonal matrix with diagonal blocks of size $(n_{j_{l+1}} - n_{j_l}) \times (n_{j_{l+1}} - n_{j_l})$, $l = 1, \dots, k-1$.

There are two fundamentally different approaches for constructing inner vectors with the property (2.20). In both cases, we generate the inner vectors using a simple three-term recurrence. However, in the first approach, each inner vector in a block is biorthogonalized against the previous block as soon as it is constructed. This variant will be called the *sequential algorithm.* In the second approach, the entire block is constructed before it is biorthogonalized against the previous block and, possibly, depending on the size of the current block, against vectors from blocks further back. This variant will be called the *block algorithm.* The sequential algorithm is more suitable for a serial computer, while the block algorithm is more suitable for a parallel computer. In this paper, we will focus on the sequential algorithm and its implementation, and we will only sketch the block algorithm.

Finally, two more notes. First, the inner product (2.11) could have been defined as

$$(\overline{\Phi}, \Psi) := \left(\overline{\Phi}(A^H) w_1\right)^H \left(\Phi(A) v_1\right) = w_1^H \Phi(A) \Psi(A) v_1$$

and the algorithms can be formulated equally well in either terms. We use the transpose because it simplifies the formulas. Second, in the rest of the paper, we will use the notation $n_k := n_{j_k}$ for the indices of the "well-defined" regular vectors. However, notice that there is no guarantee that the indices n_k generated by the look-ahead Lanczos algorithm in finite precision arithmetic actually satisfy (2.18).

3. NOTATION

In this section, we introduce some further notation.

We will use the following indices:

- $n = 1, 2, \dots$ are the indices of the Lanczos vectors v_n and w_n;
- $l = 1, 2, \dots$ is used as a counter for the blocks;
- n_l, $l = 1, 2, \dots$, $n_1 := 1$, are the indices of the computed regular vectors; note that n_l is also the index of the vector at the beginning of the lth block;
- $h_l := n_{l+1} - n_l$, $l = 1, 2, \dots$, is the size of the lth block;
- For given n, $k = k(n)$ is the number of the block which contains the Lanczos vectors v_n and w_n; note that n_k is the index of the last computed regular vector with index $\leq n$;
- ν and μ are 0-based indices used to count inside a block;
- i, j, and m are general purpose indices.

For reasons of stability, we will compute scaled versions of the right and left Lanczos vectors, rather than the "monic" vectors v_n and w_n (cf. (2.16)) corresponding to monic FOPs. A proven choice (see [18], [19]) is to scale the Lanczos vectors to have unit length. By $\hat{v}_n$ and $\hat{w}_n$ we denote the scaled versions defined by

$$v_n = s_n \hat{v}_n \quad \text{and} \quad w_n = t_n \hat{w}_n, \quad s_n := \|v_n\|, t_n := \|w_n\|. \tag{3.1}$$

The scale factors s_n and t_n in (3.1) can easily reach the overflow or underflow limits; hence, instead of storing them directly, we store $\frac{s_n}{s_{n-1}}$, $\frac{t_n}{t_{n-1}}$, and $\frac{s_n}{t_n}$, and we never compute s_n or t_n directly. Furthermore, in order to save work, the vectors will not actually be scaled at every step. Instead, we store vectors $\mathring{v}_n$ and $\mathring{w}_n$, with scale factors σ_n and ξ_n, such that

$$\hat{v}_n = \sigma_n \mathring{v}_n \quad \text{and} \quad \hat{w}_n = \xi_n \mathring{w}_n, \quad \sigma_n,\ \xi_n > 0, \tag{3.2}$$

and we actually carry out the scaling only when σ_n or ξ_n approach the overflow or underflow limits. (The scale factors σ_n are not to be confused with the singular values $\sigma_{\min}$ and $\sigma_{\max}$.)

We identify blocks by their number l. Capital letters with subscript l denote matrices whose columns are all the vectors from block l. For example,

$$V_l = [\, v_{n_l} \quad v_{n_l+1} \quad \dots \quad v_{n_{l+1}-1} \,] \quad \text{and} \quad \hat{W}_l = [\, \hat{w}_{n_l} \quad \hat{w}_{n_l+1} \quad \dots \quad \hat{w}_{n_{l+1}-1} \,]$$

are the matrices containing the "monic" right Lanczos vectors and the scaled left Lanczos vectors corresponding to block l, respectively. By S_l and T_l, we denote the diagonal matrices whose diagonal entries are the scaling factors, as defined in (3.1), corresponding to block l. Note that

$$V_l = \hat{V}_l S_l \quad \text{and} \quad W_l = \hat{W}_l T_l.$$

Similarly, Σ_l and Ξ_l denote the diagonal scaling matrices containing the scalar factors from (3.2) corresponding to block l, and then

$$\hat{V}_l = \mathring{V}_l \Sigma_l \quad \text{and} \quad \hat{W}_l = \mathring{W}_l \Xi_l.$$

Capital letters with superscripts $^{(n)}$ indicate matrices which contain quantities from all previous iterations up to step n. For example, in addition to (2.4), we denote by $\hat{V}^{(n)}$ and $\hat{W}^{(n)}$ the matrices whose columns are the first n scaled right and left Lanczos vectors, respectively; similarly, $S^{(n)}$ and $T^{(n)}$ are the diagonal matrices containing the corresponding scaling factors from (3.1). In view of (2.5) and (2.20), we then have, for $n = 1, 2, \ldots,$

$$\begin{aligned} A\hat{V}^{(n)} &= \hat{V}^{(n)} S^{(n)} H^{(n)} (S^{(n)})^{-1} + [0 \quad \cdots \quad 0 \quad \tfrac{s_{n+1}}{s_n} \hat{v}_{n+1}\,], \\ A^T \hat{W}^{(n)} &= \hat{W}^{(n)} T^{(n)} H^{(n)} (T^{(n)})^{-1} + [0 \quad \cdots \quad 0 \quad \tfrac{t_{n+1}}{t_n} \hat{w}_{n+1}\,], \end{aligned} \tag{3.3}$$

and

$$(\hat{W}^{(n)})^T \hat{V}^{(n)} = (T^{(n)})^{-1} D^{(n)} (S^{(n)})^{-1}. \tag{3.4}$$

Here

$$H^{(n)} := \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \gamma_2 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_k \\ 0 & \cdots & 0 & \gamma_k & \alpha_k \end{bmatrix} \tag{3.5}$$

is a $n \times n$ block tridiagonal matrix with blocks of the form

$$\alpha_l = \begin{bmatrix} * & \cdots & \cdots & \cdots & * \\ 1 & \ddots & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & * \end{bmatrix}, \qquad \gamma_l = \begin{bmatrix} 0 & \cdots & \cdots & 0 & 1 \\ \vdots & \ddots & & & 0 \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix}. \tag{3.6}$$

The blocks β_l are in general full matrices. Notice that $H^{(n)}$ is an upper Hessenberg matrix.

For $l < k := k(n)$ the matrices α_l, β_l, and γ_l are of size $h_l \times h_l$, $h_{l-1} \times h_l$, and $h_l \times h_{l-1}$, respectively. The matrices α_k, β_k, and γ_k corresponding to the current, *i.e.* kth, block are of size $\tilde{h}_k \times \tilde{h}_k$, $h_{k-1} \times \tilde{h}_k$, and $\tilde{h}_k \times h_{k-1}$, respectively, where $\tilde{h}_k := n + 1 - n_k$. Notice that in general the kth block is not a complete block; it is complete if, and only if, $n+1$ is the index of the next computed regular vector. In (3.4), the matrix

$$D^{(n)} = \operatorname{diag}(W_1^T V_1, W_2^T V_2, \ldots, W_k^T V_k) \tag{3.7}$$

is block diagonal with nonsingular blocks $W_l^T V_l$, $l = 1, 2, \ldots, k-1$. Its last block $W_k^T V_k$, and hence $D^{(n)}$ itself, is nonsingular if the kth block is complete.

Furthermore, the following notation will be used. We set

$$\hat{H}^{(n)} := S^{(n)} H^{(n)} (S^{(n)})^{-1} \quad \text{and} \quad M_l := (\hat{W}_l^T \hat{V}_l)^{-1} \hat{W}_l^T.$$

Generally, a ˜ (tilde), as *e.g.* in $\tilde{v}_{n+1}$, denotes intermediate quantities. $A_{:,m}$ means the mth column of A, while $A_{i:j,m}$ means elements i through j of the mth column of A.

We will assume that the vectors in a block are generated using a polynomial recursion of the form

$$\begin{aligned} \Theta_{\nu+1}(z) =& (z - \zeta_\nu)\,\Theta_\nu(z) - \eta_\nu \Theta_{\nu-1}(z), \quad \nu = 0, 1, \ldots\,, \\ &\Theta_{-1}(z) = 0, \;\; \Theta_0(z) = 1, \;\; \eta_0 = 0. \end{aligned} \tag{3.8}$$

For instance, a practical choice for the polynomials in (3.8) are suitably scaled and translated Chebyshev polynomials, so that the inner vectors are generated by the Chebyshev iteration [16]. Finally, $\Theta_\nu(A)$ will be denoted by just Θ_ν whenever the meaning is clear from the context.

4. THE SEQUENTIAL ALGORITHM

The sequential version of the algorithm biorthogonalizes each inner vector in a block against the vectors in the previous block as soon as the vector is constructed, and biorthogonalizes the regular vectors against the previous two blocks.

Suppose we have already completed n steps of the algorithm. Hence, v_n and w_n are the last generated Lanczos vectors and $k = k(n)$ is the index of the last block. If v_{n+1} and w_{n+1} are constructed as inner vectors, then they are given by

$$\begin{aligned}
\tilde{v}_{n+1} &= Av_n - \zeta_{n-n_k} v_n - \eta_{n-n_k} v_{n-1},\\
\tilde{w}_{n+1} &= A^T w_n - \zeta_{n-n_k} w_n - \eta_{n-n_k} w_{n-1},
\end{aligned}$$

$$\begin{aligned}
v_{n+1} &= \tilde{v}_{n+1} - V_{k-1}(W_{k-1}^T V_{k-1})^{-1} W_{k-1}^T \tilde{v}_{n+1}\\
&= \tilde{v}_{n+1} - V_{k-1}(W_{k-1}^T V_{k-1})^{-1} W_{k-1}^T A v_n,\\
w_{n+1} &= \tilde{w}_{n+1} - W_{k-1}(W_{k-1}^T V_{k-1})^{-1} W_{k-1}^T \tilde{v}_{n+1}\\
&= \tilde{w}_{n+1} - W_{k-1}(W_{k-1}^T V_{k-1})^{-1} W_{k-1}^T A v_n,
\end{aligned}$$

or, in terms of scaled vectors, by

$$\begin{aligned}
\frac{s_{n+1}}{s_n}\hat{v}_{n+1} &= A\hat{v}_n - \zeta_{n-n_k}\hat{v}_n - \frac{s_{n-1}}{s_n}\eta_{n-n_k}\hat{v}_{n-1}\\
&\quad - \hat{V}_{k-1}(\hat{W}_{k-1}^T \hat{V}_{k-1})^{-1}\hat{W}_{k-1}^T A\hat{v}_n,\\
\frac{t_{n+1}}{t_n}\hat{w}_{n+1} &= A^T\hat{w}_n - \zeta_{n-n_k}\hat{w}_n - \frac{t_{n-1}}{t_n}\eta_{n-n_k}\hat{w}_{n-1}\\
&\quad - \frac{s_n}{t_n}\hat{W}_{k-1}T_{k-1}S_{k-1}^{-1}(\hat{W}_{k-1}^T \hat{V}_{k-1})^{-1}\hat{W}_{k-1}^T A\hat{v}_n.
\end{aligned}$$

If v_{n+1} and w_{n+1} are regular vectors, then they are given by

$$\begin{aligned}
\tilde{v}_{n+1} &= Av_n,\\
\tilde{w}_{n+1} &= A^T w_n,
\end{aligned}$$

$$\begin{aligned}
v_{n+1} &= \tilde{v}_{n+1} - V_{k-1}(W_{k-1}^T V_{k-1})^{-1} W_{k-1}^T \tilde{v}_{n+1} - V_k(W_k^T V_k)^{-1} W_k^T \tilde{v}_{n+1}\\
&= \tilde{v}_{n+1} - V_{k-1}(W_{k-1}^T V_{k-1})^{-1} W_{k-1}^T A v_n - V_k(W_k^T V_k)^{-1} W_k^T A v_n,\\
w_{n+1} &= \tilde{w}_{n+1} - W_{k-1}(W_{k-1}^T V_{k-1})^{-1} W_{k-1}^T \tilde{v}_{n+1} - W_k(W_k^T V_k)^{-1} W_k^T \tilde{v}_{n+1}\\
&= \tilde{w}_{n+1} - W_{k-1}(W_{k-1}^T V_{k-1})^{-1} W_{k-1}^T A v_n - W_k(W_k^T V_k)^{-1} W_k^T A v_n,
\end{aligned}$$

or, in terms of scaled vectors, by

$$\begin{aligned}
\frac{s_{n+1}}{s_n}\hat{v}_{n+1} &= A\hat{v}_n \\
&\quad - \hat{V}_{k-1}(\hat{W}_{k-1}^T\hat{V}_{k-1})^{-1}\hat{W}_{k-1}^T A\hat{v}_n && (4.1a)\\
&\quad - \hat{V}_k(\hat{W}_k^T\hat{V}_k)^{-1}\hat{W}_k^T A\hat{v}_n, && (4.1b)\\
\frac{t_{n+1}}{t_n}\hat{w}_{n+1} &= A^T\hat{w}_n \\
&\quad - \frac{s_n}{t_n}\hat{W}_{k-1}T_{k-1}S_{k-1}^{-1}(\hat{W}_{k-1}^T\hat{V}_{k-1})^{-1}\hat{W}_{k-1}^T A\hat{v}_n && (4.2a)\\
&\quad - \frac{s_n}{t_n}\hat{W}_k T_k S_k^{-1}(\hat{W}_k^T\hat{V}_k)^{-1}\hat{W}_k^T A\hat{v}_n. && (4.2b)
\end{aligned}$$

Here we used the fact that at step n, the inner vectors v_n and (when appropriate) v_{n-1} are already biorthogonal to the previous block W_{k-1}. Note that using the recursion to compute $\tilde{v}_{n+1}$ and $\tilde{w}_{n+1}$ in the case of the regular vectors is redundant, since the regular vectors are then biorthogonalized against the vectors in block k, which includes the vectors involved in the recursion.

If v_{n+1} and w_{n+1} are inner vectors, the size of the current incomplete block k is increased by 1; if they are regular vectors, then the kth block is complete, and a new block, the $(k+1)$st is started with v_{n+1} and w_{n+1} as its first vectors. The decision on whether to construct v_{n+1} and w_{n+1} as inner or as regular vectors is based on three different criteria, see (4.10–4.12). If at least one is satisfied, then v_{n+1} and w_{n+1} are constructed as inner vectors, otherwise, they are constructed as regular vectors. Next, we motivate the three criteria.

First, recall (cf. (3.7)) that v_{n+1} and w_{n+1} are regular vectors if, and only if, $W_k^T V_k$ is nonsingular. Therefore, we check whether this matrix is singular or close to singular. The singular value decomposition (SVD) of $\hat{W}_k^T\hat{V}_k$ is computed, and an inner step is performed if

$$\sigma_{\min}(\hat{W}_k^T\hat{V}_k) < tol. \tag{4.3}$$

Here tol is a suitably chosen tolerance. For example, Parlett [17] suggests $tol = \epsilon^{1/4}$ or $tol = \epsilon^{1/3}$, where ϵ denotes the roundoff unit. In view of (4.3), it is guaranteed that complete blocks of constructed Lanczos vectors satisfy

$$\sigma_{\min}(\hat{W}_l^T\hat{V}_l) \geq tol, \quad l = 1, 2, \ldots\ . \tag{4.4}$$

Note that, by [17, Theorem 10.1], (4.4) together with (3.4) and (3.7) imply

$$\sigma_{\min}(\hat{V}^{(n)}) \geq \frac{tol}{\sqrt{n}} \quad \text{and} \quad \sigma_{\min}(\hat{W}^{(n)}) \geq \frac{tol}{\sqrt{n}}, \quad n = n_l - 1,\ l = 1, 2, \ldots\ . \tag{4.5}$$

Furthermore, for the vectors corresponding to each block, we have

$$\sigma_{\min}(\hat{V}_l) \geq \frac{tol}{\sqrt{h_l}} \quad \text{and} \quad \sigma_{\min}(\hat{W}_l) \geq \frac{tol}{\sqrt{h_l}}, \quad l = 1, 2, \ldots .$$

Remark that the columns of $\hat{V}^{(n)}$ and $\hat{W}^{(n)}$ are unit vectors and that $\sigma_{\min}(\hat{V})$ respectively $\sigma_{\min}(\hat{W})$ is a measure of the linear independence of these vectors. In particular, (4.5) ensures that the Lanczos vectors remain linearly independent.

In the outlined algorithm, the block biorthogonality (3.4) and (3.7) is enforced only between two or three successive blocks. Unfortunately, in finite precision arithmetic, biorthogonality of blocks whose indices are far apart is typically lost. Consequently, in practice, (4.5) is no longer guaranteed to hold and thus (4.4) alone does not ensure that the computed Lanczos vectors are sufficiently linearly independent. Indeed, numerical tests confirmed that, if the look-ahead strategy is based only on criterion (4.3), the algorithm may produce within a block Lanczos vectors which are almost linearly dependent. When this happens, the algorithm never completes the current block, *i.e.* it has generated an "artificial" incurable breakdown.

The situation just described occurs if, roughly speaking, a regular vector v_{n+1} is computed whose component $Av_n \in K_{n+1}(v_1, A)$ is dominated by its component in the previous Krylov space $K_n(v_1, A)$ (and similarly for w_{n+1}). In order to avoid the construction of such regular vectors, we check the l_1-norm of the coefficients for $\hat{V}_{k-1}$ in (4.1a) and $\hat{V}_k$ in (4.1b) and compute v_{n+1} as a regular vector only if

$$\sum_{j=n_{k-1}}^{n_k-1} \left| ((\hat{W}_{k-1}^T \hat{V}_{k-1})^{-1} \hat{W}_{k-1}^T A \hat{v}_n)_j \right| \leq fac \cdot \|A\| \tag{4.6}$$

and

$$\sum_{j=n_k}^{n} \left| ((\hat{W}_k^T \hat{V}_k)^{-1} \hat{W}_k^T A \hat{v}_n)_j \right| \leq fac \cdot \|A\| . \tag{4.7}$$

Here fac is a suitably chosen factor. In analogy, by (4.2a) and (4.2b), w_{n+1} is constructed as a regular vector only if

$$\frac{s_n}{t_n} \sum_{j=n_{k-1}}^{n_k-1} \frac{t_j}{s_j} \left| ((\hat{W}_{k-1}^T \hat{V}_{k-1})^{-1} \hat{W}_{k-1}^T A \hat{v}_n)_j \right| \leq fac \cdot \|A\| \tag{4.8}$$

and

$$\frac{s_n}{t_n} \sum_{j=n_k}^{n} \frac{t_j}{s_j} \left| ((\hat{W}_k^T \hat{V}_k)^{-1} \hat{W}_k^T A \hat{v}_n)_j \right| \leq fac \cdot \|A\| . \tag{4.9}$$

By combining (4.7) and (4.9), we arrive at the criterion (4.11) given below for performing an inner step. Notice that (4.6) and (4.8) involve quantities from the previous block $k-1$. Hence, if (4.6) or (4.8) were violated, one would need to go back and construct the previous regular vectors v_{n_k} and w_{n_k} and the kth block differently. In order to avoid this, we check for (4.6) and (4.8) while building block $k-1$, which results in criterion (4.12) below for performing an inner step.

To summarize, we next describe one step of the sequential algorithm.

DESCRIPTION OF ONE STEP OF THE SEQUENTIAL ALGORITHM:

Given $\mathring{v}_n$, $\mathring{w}_n$, σ_n, ξ_n, $\frac{s_n}{s_{n-1}}$, $\frac{t_n}{t_{n-1}}$, $\frac{s_n}{t_n}$, $k=k(n)$, $(\hat{W}_k^T\hat{V}_k)$, and $\hat{H}_{n_{k-1}:n_k-1,n}$.

Compute $\tilde{v}_{n+1} = A\mathring{v}_n$, $\tilde{w}_{n+1} = A^T\mathring{w}_n$.

Compute

$$\tilde{v}_{n+1} = \tilde{v}_{n+1} - \frac{1}{\sigma_n}\mathring{V}_{k-1}\Sigma_{k-1}\hat{H}_{n_{k-1}:n_k-1,n},$$

$$\tilde{w}_{n+1} = \tilde{w}_{n+1} - \frac{1}{\xi_n}\frac{s_n}{t_n}\mathring{W}_{k-1}\Xi_{k-1}T_{k-1}S_{k-1}^{-1}\hat{H}_{k-1:n_k-1,n}.$$

Compute $\mathring{w}_n^T\tilde{v}_{n+1}$.

Compute the SVD of $(\hat{W}_k^T\hat{V}_k)$.

Set

$$\text{inner} = \left(\sigma_{\min}(\hat{W}_k^T\hat{V}_k)\right) < tol. \tag{4.10}$$

If *not* inner, then

Call **BUILDH1** to compute $\tilde{H}_{n_k:n,n} = (\hat{W}_k^T\hat{V}_k)^{-1}\hat{W}_k^T A\hat{v}_n$.

Set

$$\text{inner} = \left(\max\left\{\sum_{j=n_k}^{n}\left|\tilde{H}_{j,n}\right|,\ \frac{s_n}{t_n}\sum_{j=n_k}^{n}\frac{t_j}{s_j}\left|\tilde{H}_{j,n}\right|\right\}\right) > fac\cdot\|A\|. \tag{4.11}$$

If *not* inner, then

Build v_{n+1} and w_{n+1} as regular vectors:

$$\frac{s_{n+1}}{s_n}\left(\sigma_{n+1}\mathring{v}_{n+1}\right) = \sigma_n\left(\tilde{v}_{n+1} - \frac{1}{\sigma_n}\mathring{V}_k\Sigma_k\tilde{H}_{n_k:n,n}\right),$$

$$\frac{t_{n+1}}{t_n}\left(\xi_{n+1}\mathring{w}_{n+1}\right) = \xi_n\left(\tilde{w}_{n+1} - \frac{1}{\xi_n}\frac{s_n}{t_n}\mathring{W}_k \Xi_k T_k S_k^{-1}\tilde{H}_{n_k:n,n}\right).$$

```
Call SCALE to scale the vectors.
Compute $\mathring{w}_{n+1}^T\mathring{v}_{n+1}$.
Call BUILDH2 to compute
$\tilde{H}_{n_k:n,n+1} = (\hat{W}_k^T\hat{V}_k)^{-1}\hat{W}_k^T A\hat{v}_{n+1}$.
Set
```

$$\texttt{inner} = \left(\max\left\{\sum_{j=n_k}^{n}\left|\tilde{H}_{j,n+1}\right|,\ \frac{s_{n+1}}{t_{n+1}}\sum_{j=n_k}^{n}\frac{t_j}{s_j}\left|\tilde{H}_{j,n+1}\right|\right\}\right) > fac\cdot\|A\|. \qquad (4.12)$$

```
If inner, then
    Build $v_{n+1}$ and $w_{n+1}$ as inner vectors:
```

$$\begin{aligned}
\frac{s_{n+1}}{s_n}\left(\sigma_{n+1}\mathring{v}_{n+1}\right) &= \sigma_n\left(\tilde{v}_{n+1} - \zeta_{n-n_k}\mathring{v}_n - \frac{s_{n-1}}{s_n}\frac{\sigma_{n-1}}{\sigma_n}\eta_{n-n_k}\mathring{v}_{n-1}\right),\\
\frac{t_{n+1}}{t_n}\left(\xi_{n+1}\mathring{w}_{n+1}\right) &= \xi_n\left(\tilde{w}_{n+1} - \zeta_{n-n_k}\mathring{w}_n - \frac{t_{n-1}}{t_n}\frac{\xi_{n-1}}{\xi_n}\eta_{n-n_k}\mathring{w}_{n-1}\right),\\
\hat{H}_{n-1,n} &= \frac{s_{n-1}}{s_n}\eta_{n-n_k},\\
\hat{H}_{n,n} &= \zeta_{n-n_k}.
\end{aligned}$$

```
    Call SCALE to scale the vectors.
    Compute $\mathring{w}_{n+1}^T\mathring{v}_{n+1}$.
    Set $k = k(n+1) = k(n)$.
    Call UPDATE to update $(\hat{W}_k^T\hat{V}_k)$.
    Call BUILDH2 to compute $\hat{H}_{n_{k-1}:n_k-1,n+1} = (\hat{W}_{k-1}^T\hat{V}_{k-1})^{-1}\hat{W}_{k-1}^T A\hat{v}_{n+1}$.
else
    Set $\hat{H}_{n_k:n,n} = \tilde{H}_{n_k:n,n}$ and $\hat{H}_{n_k:n,n+1} = \tilde{H}_{n_k:n,n+1}$.
    Set $k = k(n+1) = k(n)+1$.
    Set $(\hat{W}_k^T\hat{V}_k) = \sigma_{n+1}\xi_{n+1}\mathring{w}_{n+1}^T\mathring{v}_{n+1}$.
Set $\hat{H}_{n+1,n} = \frac{s_{n+1}}{s_n}$.
```

5. IMPLEMENTATION DETAILS

In this section, we describe in detail the actual implementation of the routines **BUILDH1**, **BUILDH2**, **SCALE**, and **UPDATE** used in the sequential algorithm, as well as the procedure for estimating fac (cf. 4.6–4.9). Note that in the actual codes, these routines — except for **SCALE** — do not appear explicitly; rather, they are coded inline, and appear only as logical blocks.

5.1. BUILDH1

BUILDH1 takes as input $(\hat{W}_k^T\hat{V}_k)_{:,n-n_k+1}$, $(\hat{W}_k^T\hat{V}_k)^{-1}$, and $\sigma_n\xi_n\mathring{w}_n^T\tilde{v}_{n+1}$, and returns $\tilde{H}_{n_k:n,n}$.

Consider a term of the form $w_j^TAv_n$, $n_k \le j \le n$, $n = n_{k+1}-1$. Let $\nu = j - n_k$, $\mu = n - n_k$. We distinguish two cases:

(i) $j = n$.

We compute the $w_n^TAv_n$ term directly, since we do not know all the terms in the recurrence for either vector.

(ii) $n_k \le j \le n-1$.

Here,

$$\begin{aligned}
w_j^TAv_{n_{k+1}-1} &= (\Theta_\nu(A^T)w_{n_k})^TA(\Theta_\mu(A)v_{n_k}) \\
&= w_{n_k}^T\Theta_\nu A\Theta_\mu v_{n_k} \\
&= w_{n_k}^T A\Theta_\nu\Theta_\mu v_{n_k} \\
&= w_{n_k}^T(\Theta_{\nu+1}+\zeta_\nu\Theta_\nu+\eta_\nu\Theta_{\nu-1})\Theta_\mu v_{n_k} \\
&= w_{n_k}^T\Theta_{\nu+1}\Theta_\mu v_{n_k}+\zeta_\nu w_{n_k}^T\Theta_\nu\Theta_\mu v_{n_k}+\eta_\nu w_{n_k}^T\Theta_{\nu-1}\Theta_\mu v_{n_k} \\
&= w_{j+1}^Tv_n+\zeta_\nu w_j^Tv_n+\eta_\nu w_{j-1}^Tv_n,
\end{aligned}$$

all of which are terms which have already been computed as part of $W_k^TV_k$. Hence, we have:

$$\begin{aligned}
\hat{W}_k^TA\hat{v}_n &= (W_kT_k^{-1})^TA\hat{v}_n \\
&= T_k^{-T}[\,w_{n_k}^TA\hat{v}_n \quad \cdots \quad w_{n-1}^TA\hat{v}_n \quad w_n^TA\hat{v}_n\,]^T \\
&= T_k^{-T}[\cdots \quad (w_{j+1}+\zeta_\nu w_j+\eta_\nu w_{j-1})^T\hat{v}_n \quad \cdots \quad w_n^TA\hat{v}_n\,]^T \\
&= T_k^{-T}[\cdots \quad w_{j+1}^T\hat{v}_n+\zeta_\nu w_j^T\hat{v}_n+\eta_\nu w_{j-1}^T\hat{v}_n \quad \cdots \quad w_n^TA\hat{v}_n\,]^T
\end{aligned}$$

$$= T_k^{-T} [\cdots \quad (t_{j+1}\hat{w}_{j+1}^T + \zeta_\nu t_j \hat{w}_j^T + \eta_\nu t_{j-1}\hat{w}_{j-1}^T)\hat{v}_n \quad \cdots \quad t_n \hat{w}_n^T A \hat{v}_n]^T$$
$$= \left[\cdots \quad \frac{t_{j+1}}{t_j}\hat{w}_{j+1}^T\hat{v}_n + \zeta_\nu \hat{w}_j^T \hat{v}_n + \eta_\nu \frac{t_{j-1}}{t_j}\hat{w}_{j-1}^T\hat{v}_n \quad \cdots \quad \hat{w}_n^T A \hat{v}_n \right]^T$$

where $j = n_k, \ldots, n_{k+1} - 2$. With this, $\tilde{H}_{n_k:n,n} = (\hat{W}_k^T \hat{V}_k)^{-1} \hat{W}_k^T A \hat{v}_n$. For the $\hat{w}_n^T A \hat{v}_n$ term, we use:

$$\begin{aligned} \hat{w}_n^T A \hat{v}_n &= \hat{w}_n^T (A\hat{v}_n - \hat{V}_{k-1}\hat{H}_{n_{k-1}:n_k-1,n}) \\ &= \hat{w}_n^T (\sigma_n \tilde{v}_{n+1}) \\ &= \sigma_n \xi_n \mathring{w}_n^T \tilde{v}_{n+1} \end{aligned}$$

because this formulation has better numerical properties.

5.2. BUILDH2

The logical block **BUILDH2** takes as input σ_{n+1}, ξ_{n_k}, $\frac{t_{n_k}}{t_{n_k-1}}$, $\mathring{w}_{n_k}^T\mathring{v}_{n+1}$, and the last column of $(\hat{W}_{k-1}^T\hat{V}_{k-1})^{-1}$, and returns $\hat{H}_{n_{k-1}:n_k-1,n+1}$.

Consider a term of the form $w_j^TAv_{n+1}$, $n_{k-1}\le j\le n_k-1$, $n_k-1\le n\le n_{k+1}-2$. Let $\nu=j-n_{k-1}$, $\mu=n-n_k$. We distinguish two cases:

(i) $j=n_k-1$.

$$\begin{aligned} w_j^TAv_{n+1} &= (\Theta_\nu(A^T)w_{n_{k-1}})^TAv_{n+1}\\ &= w_{n_{k-1}}^T\Theta_\nu Av_{n+1}\\ &= w_{n_{k-1}}^TA\Theta_\nu v_{n+1}\\ &= w_{n_k}^Tv_{n+1}. \end{aligned}$$

Here we used the fact that the polynomial for v_{n+1} is orthogonal to the polynomials in blocks n_{k-1} and n_{k-2}, which appear in the expression for the polynomial for w_{n_k}.

(ii) $n_{k-1}\le j\le n_k-2$.

$$\begin{aligned} w_j^TAv_{n+1} &= (\Theta_\nu(A^T)w_{n_{k-1}})^TAv_{n+1}\\ &= w_{n_{k-1}}^T\Theta_\nu Av_{n+1}\\ &= w_{n_{k-1}}^TA\Theta_\nu v_{n+1}\\ &= w_{n_{k-1}}^T(\Theta_{\nu+1}+\zeta_\nu\Theta_\nu+\eta_\nu\Theta_{\nu-1})v_{n+1}\\ &= 0. \end{aligned}$$

Hence, we have:

$$\begin{aligned} \hat{H}_{n_{k-1}:n_k-1,n+1} &= (\hat{W}_{k-1}^T\hat{V}_{k-1})^{-1}\hat{W}_{k-1}^TA\hat{v}_{n+1}\\ &= (\hat{W}_{k-1}^T\hat{V}_{k-1})^{-1}(W_{k-1}T_{k-1}^{-1})^TA\hat{v}_{n+1}\\ &= (\hat{W}_{k-1}^T\hat{V}_{k-1})^{-1}T_{k-1}^{-T}\\ &\quad\cdot\begin{bmatrix} w_{n_{k-1}}^TA\hat{v}_{n+1} & \cdots & w_{n_k-2}^TA\hat{v}_{n+1} & w_{n_k-1}^TA\hat{v}_{n+1}\end{bmatrix}^T \end{aligned}$$

$$
\begin{aligned}
&= (\hat{W}_{k-1}^T \hat{V}_{k-1})^{-1} T_{k-1}^{-T} \begin{bmatrix} 0 & \cdots & 0 & w_{n_k}^T \hat{v}_{n+1} \end{bmatrix}^T \\
&= (\hat{W}_{k-1}^T \hat{V}_{k-1})^{-1} T_{k-1}^{-T} \begin{bmatrix} 0 & \cdots & 0 & (t_{n_k} \hat{w}_{n_k})^T \hat{v}_{n+1} \end{bmatrix}^T \\
&= (\hat{W}_{k-1}^T \hat{V}_{k-1})^{-1} \begin{bmatrix} 0 & \cdots & 0 & \frac{t_{n_k}}{t_{n_k-1}} \hat{w}_{n_k}^T \hat{v}_{n+1} \end{bmatrix}^T \\
&= (\hat{W}_{k-1}^T \hat{V}_{k-1})^{-1} \begin{bmatrix} 0 & \cdots & 0 & \frac{t_{n_k}}{t_{n_k-1}} \sigma_{n+1} \xi_{n_k} \mathring{w}_{n_k}^T \mathring{v}_{n+1} \end{bmatrix}^T \\
&= \sigma_{n+1} \xi_{n_k} \frac{t_{n_k}}{t_{n_k-1}} \mathring{w}_{n_k}^T \mathring{v}_{n+1} (\hat{W}_{k-1}^T \hat{V}_{k-1})_{:,h_{k-1}}^{-1}.
\end{aligned}
$$

5.3. SCALE

SCALE takes as input σ_n, ξ_n, $\tilde{v}_{n+1}$, $\tilde{w}_{n+1}$, $\frac{s_n}{t_n}$, and returns $\mathring{v}_{n+1}$, $\mathring{w}_{n+1}$, σ_{n+1}, ξ_{n+1}, $\frac{s_{n+1}}{s_n}$, $\frac{t_{n+1}}{t_n}$, and $\frac{s_{n+1}}{t_{n+1}}$. It computes the scale factors s_{n+1} and t_{n+1} so that $\hat{v}_{n+1}$ and $\hat{w}_{n+1}$ both are unit vectors. This gives:

$$\left(\frac{s_{n+1}}{s_n}\right)\left(\sigma_{n+1}\mathring{v}_{n+1}\right) = \left(\sigma_n \left\|\tilde{v}_{n+1}\right\|\right)\left(\frac{1}{\left\|\tilde{v}_{n+1}\right\|}\tilde{v}_{n+1}\right),$$
$$\left(\frac{t_{n+1}}{t_n}\right)\left(\xi_{n+1}\mathring{w}_{n+1}\right) = \left(\xi_n \left\|\tilde{w}_{n+1}\right\|\right)\left(\frac{1}{\left\|\tilde{w}_{n+1}\right\|}\tilde{w}_{n+1}\right).$$

The algorithm is as follows:

```
        $s = \|\tilde{v}_{n+1}\|$,
        $t = \|\tilde{w}_{n+1}\|$.
            If either $s \cdot \sigma_n$ or $t \cdot \xi_n$ is small, we have found an invariant subspace.
$\sigma_{n+1} = \frac{1}{s}$,
$\xi_{n+1} = \frac{1}{t}$.
            If $\sigma_{n+1}$ is too small or too large, then
              $\tilde{v}_{n+1} = \sigma_{n+1}\tilde{v}_{n+1}$,
              $\sigma_{n+1} = 1$.
            If $\xi_{n+1}$ is too small or too large, then
              $\tilde{w}_{n+1} = \xi_{n+1}\tilde{w}_{n+1}$,
              $\xi_{n+1} = 1$,
$\frac{s_{n+1}}{s_n} = \sigma_n \|\tilde{v}_{n+1}\|$,
$\frac{t_{n+1}}{t_n} = \xi_n \|\tilde{w}_{n+1}\|$,
$\frac{s_{n+1}}{t_{n+1}} = \frac{s_{n+1}}{s_n}\frac{t_n}{t_{n+1}}\frac{s_n}{t_n}$,
$\mathring{v}_{n+1} = \tilde{v}_{n+1}$,
$\mathring{w}_{n+1} = \tilde{w}_{n+1}$.
```

5.4. UPDATE

UPDATE takes as input σ_{n+1}, ξ_{n+1}, $\mathring{w}_{n+1}^T \mathring{v}_{n+1}$, and $(\hat{W}_k^T \hat{V}_k)$. It appends a new row and column to the matrix $(\hat{W}_k^T \hat{V}_k)$. First note that $(W_k^T V_k)$ is symmetric, hence only its upper triangular part has to be constructed. Let w_i and v_j be two vectors from the current block, and $\nu = i - n_k$, $\mu = j - n_k$ be the corresponding block indices.

Then,

$$
\begin{aligned}
w_i^T v_j &= \left(\Theta_\nu(A^T) w_{n_k}\right)^T \Theta_\mu(A) v_{n_k} \\
&= w_{n_k}^T \Theta_\nu \Theta_\mu v_{n_k} \\
&= w_{n_k}^T \Theta_\nu (A\Theta_{\mu-1} - \zeta_{\mu-1}\Theta_{\mu-1} - \eta_{\mu-1}\Theta_{\mu-2}) v_{n_k} \\
&= w_{n_k}^T \Theta_\nu A \Theta_{\mu-1} v_{n_k} - w_{n_k}^T (\zeta_{\mu-1}\Theta_\nu\Theta_{\mu-1} + \eta_{\mu-1}\Theta_\nu\Theta_{\mu-2}) v_{n_k} \\
&= w_{n_k}^T A \Theta_\nu \Theta_{\mu-1} v_{n_k} - w_{n_k}^T (\zeta_{\mu-1}\Theta_\nu\Theta_{\mu-1} + \eta_{\mu-1}\Theta_\nu\Theta_{\mu-2}) v_{n_k} \\
&= w_{n_k}^T (\Theta_{\nu+1} + \zeta_\nu \Theta_\nu + \eta_\nu \Theta_{\nu-1}) \Theta_{\mu-1} v_{n_k} \\
&\quad - w_{n_k}^T (\zeta_{\mu-1}\Theta_\nu\Theta_{\mu-1} + \eta_{\mu-1}\Theta_\nu\Theta_{\mu-2}) v_{n_k} \\
&= w_{n_k}^T \Theta_{\nu+1}\Theta_{\mu-1} v_{n_k} + \zeta_\nu w_{n_k}^T \Theta_\nu \Theta_{\mu-1} v_{n_k} + \eta_\nu w_{n_k}^T \Theta_{\nu-1}\Theta_{\mu-1} v_{n_k} \\
&\quad - \zeta_{\mu-1} w_{n_k}^T \Theta_\nu \Theta_{\mu-1} v_{n_k} - \eta_{\mu-1} w_{n_k}^T \Theta_\nu \Theta_{\mu-2} v_{n_k} \\
&= w_{n_k}^T (\Theta_{\nu+1}\Theta_{\mu-1} - \eta_{\mu-1}\Theta_\nu\Theta_{\mu-2}) v_{n_k} + (\zeta_\nu - \zeta_{\mu-1}) w_{n_k}^T \Theta_\nu \Theta_{\mu-1} v_{n_k} \\
&\quad + \eta_\mu w_{n_k}^T \Theta_{\nu-1}\Theta_{\mu-1} v_{n_k}. \\
&= w_{i+1}^T v_{j-1} - \eta_{j-n_k-1} w_i^T v_{j-2} + (\zeta_{i-n_k} - \zeta_{j-n_k-1}) w_i^T v_{j-1} \\
&\quad + \eta_{j-n_k} w_{i-1}^T v_{j-1}.
\end{aligned}
$$

This shows that an element on the mth diagonal can be computed from elements on the previous two diagonals and previous elements on the mth diagonal, leaving only the main diagonal and the first superdiagonal to be computed by inner products. Hence, the complete $h_k \times h_k$ matrix $(W_k^T V_k)$ corresponding to the complete kth block can be built with only $2h_k - 1$ inner products.

To update $(\hat{W}_k^T \hat{V}_k)$, we then have:

$$\hat{w}_{n+1}^T \hat{v}_{n+1} = \sigma_{n+1}\xi_{n+1} \mathring{w}_{n+1}^T \mathring{v}_{n+1},$$

$$\hat{w}_n^T \hat{v}_{n+1} = \hat{w}_n^T \left(\frac{s_n}{s_{n+1}} \sigma_n \left(\tilde{v}_{n+1} - \zeta_{n-n_k} \mathring{v}_n - \frac{s_{n-1}}{s_n} \frac{\sigma_{n-1}}{\sigma_n} \eta_{n-n_k} \mathring{v}_{n-1} \right) \right)$$

$$
\begin{aligned}
&= \sigma_n \frac{s_n}{s_{n+1}} \left(\hat{w}_n^T \tilde{v}_{n+1} - \zeta_{n-n_k} \hat{w}_n^T \mathring{v}_n - \frac{s_{n-1}}{s_n} \frac{\sigma_{n-1}}{\sigma_n} \eta_{n-n_k} \hat{w}_n^T \mathring{v}_{n-1} \right) \\
&= \sigma_n \xi_n \frac{s_n}{s_{n+1}} \mathring{\hat{w}}_n^T \tilde{v}_{n+1} - \frac{s_n}{s_{n+1}} \zeta_{n-n_k} \hat{w}_n^T \hat{v}_n - \frac{s_n}{s_{n+1}} \frac{s_{n-1}}{s_n} \eta_{n-n_k} \hat{w}_n^T \hat{v}_{n-1},
\end{aligned}
$$

$$
\begin{aligned}
\hat{w}_i^T \hat{v}_{n+1} &= \frac{1}{s_{n+1} t_i} w_i^T v_{n+1} \\
&= \frac{1}{s_{n+1} t_i} \left(w_{i+1}^T v_n - \eta_{n-n_k} w_i^T v_{n-1} + (\zeta_{i-n_k} - \zeta_{n-n_k}) w_i^T v_n \right) \\
&\quad + \frac{1}{s_{n+1} t_i} \left(\eta_{n-n_k+1} w_{i-1}^T v_n \right) \\
&= \frac{1}{s_{n+1} t_i} \left(s_n t_{i+1} \hat{w}_{i+1}^T \hat{v}_n - \eta_{n-n_k} s_{n-1} t_i \hat{w}_i^T \hat{v}_{n-1} \right) \\
&\quad + \frac{1}{s_{n+1} t_i} \left((\zeta_{i-n_k} - \zeta_{n-n_k}) s_n t_i \hat{w}_i^T \hat{v}_n + \eta_{n-n_k+1} s_n t_{i-1} \hat{w}_{i-1}^T \hat{v}_n \right) \\
&= \frac{s_n}{s_{n+1}} \frac{t_{i+1}}{t_i} \hat{w}_{i+1}^T \hat{v}_n - \frac{s_{n-1}}{s_{n+1}} \eta_{n-n_k} \hat{w}_i^T \hat{v}_{n-1} \\
&\quad + \frac{s_n}{s_{n+1}} (\zeta_{i-n_k} - \zeta_{n-n_k}) \hat{w}_i^T \hat{v}_n + \frac{s_n}{s_{n+1}} \frac{t_{i-1}}{t_i} \eta_{n-n_k+1} \hat{w}_{i-1}^T \hat{v}_n,
\end{aligned}
$$

$$
\begin{aligned}
\hat{w}_{n+1}^T \hat{v}_i &= \frac{1}{s_i t_{n+1}} w_{n+1}^T v_i \\
&= \frac{1}{s_i t_{n+1}} w_i^T v_{n+1} \\
&= \frac{s_{n+1}}{s_i} \frac{t_i}{t_{n+1}} \hat{w}_i^T \hat{v}_{n+1},
\end{aligned}
$$

where $i = n_k, \ldots, n$.

5.5 Estimating fac

Recall that the checks (4.6–4.9) are used to ensure that the Lanczos vectors are sufficiently linearly independent to avoid artificial incurable breakdowns. However, numerical experience with matrices whose norm is known indicates that setting $fac = 1$ is too strict and results in artificial incurable breakdowns. A better setting seems to be $fac = 10$, but even this is dependent on the matrix. In addition to estimating fac, in practice one is faced with the problem of estimating the matrix norm as well. This problem becomes even more complicated when solving linear systems, because one usually replaces the original system by a preconditioned one. Finally, in practice there is also the issue of a maximal block size, which is a user-specified value related to limits on available storage. To solve the problems of estimating norms and a suitable factor fac, as well as coping with limited storage and yet allowing the algorithm to proceed as far as possible, we propose the following procedure.

Suppose we arbitrarily set $\|A\| = 1$, where A denotes the matrix actually used in generating the Lanczos vectors, thus including the case when we are solving a preconditioned linear system. Then we are left with estimating just fac, which is done dynamically. In each block, whenever an inner vector is built due to (4.11) or (4.12), the algorithm keeps track of the size of the terms that have caused (4.11) or (4.12) to be true. If the block closes, then this information is discarded. If, however, the algorithm is about to run out of storage, then fac is replaced with the smallest value which has caused an inner vector to be built, and the block is rebuilt. This time, the updated value of fac is guaranteed to pass at least once the checks in (4.11) and (4.12), and hence the block is guaranteed to close. This frees up the storage that was used by the previous block, thus guaranteeing that the algorithm can proceed (the procedure extends easily to the case when the current block is the first one).

This procedure allows then the algorithm to run until a block is built entirely due to (4.10). This situation represents an incurable breakdown, given the limits on storage, and forces the algorithm to stop.

6. THE BLOCK ALGORITHM

The block version of the algorithm differs from the sequential algorithm in that it generates the entire block before biorthogonalizing it against the previous block. This makes it more efficient on a parallel machine. For **INBLOCK**, we then have

$$\begin{aligned}\tilde{v}_{n+1} &= s_n A\hat{v}_n - s_n\zeta_{n-n_k}\hat{v}_n - s_{n-1}\eta_{n-n_k}\hat{v}_{n-1},\\ \tilde{w}_{n+1} &= t_n A^T\hat{w}_n - t_n\zeta_{n-n_k}\hat{w}_n - t_{n-1}\eta_{n-n_k}\hat{w}_{n-1}.\end{aligned}$$

In addition, depending on the recursion used, one might want to monitor the norms of the vectors and scale when needed, to ensure numerical stability. We now want to biorthogonalize the new block against the previous vectors. Let j denote the index of vector v_j in the block, $v_j = \Theta_\mu(A)\,v_{n_k}$, $\mu = j - n_k$. We want

$$[w_1 \quad \cdots \quad w_{n_k-\mu-1} \quad w_{n_k-\mu} \quad \cdots \quad w_{n_k-1}]^T v_j = 0.$$

We have

$$\begin{aligned}&[w_1 \quad \cdots \quad w_{n_k-\mu-1} \quad w_{n_k-\mu} \quad \cdots \quad w_{n_k-1}]^T v_j\\ &\quad= [w_1 \quad \cdots \quad w_{n_k-\mu-1} \quad w_{n_k-\mu} \quad \cdots \quad w_{n_k-1}]^T \Theta_\mu(A)\,v_{n_k}\\ &\quad= [\Theta_\mu(A^T)\,w_1 \quad \cdots \quad \Theta_\mu(A^T)\,w_{n_k-\mu-1} \quad \Theta_\mu(A^T)\,w_{n_k-\mu} \quad \cdots \quad \Theta_\mu(A^T)\,w_{n_k-1}]^T v_{n_k}\\ &\quad= [0 \quad \cdots \quad 0 \quad * \quad \cdots \quad *]^T,\end{aligned}$$

where the last μ entries, denoted by $*$, are generally non-zero. On one hand, the regular polynomial for v_{n_k} is orthogonal to all polynomials of degree less than n_k; on the other hand, multiplying the polynomials for the previous vectors by Θ_μ raises their degree by μ, thus raising the degree of the last μ polynomials to n_k or more, thus introducing the non-zero entries. Furthermore, in biorthogonalizing v_j against $w_{n_k-\mu}$, one introduces components along the other vectors in $v_{n_k-\mu}$'s block, as $v_{n_k-\mu}$ is not biorthogonal to the vectors in its own block. Hence, to biorthogonalize v_j against the previous vectors, we need to biorthogonalize it against all the vectors in the blocks containing $v_{n_k-\mu}$ to v_{n_k-1}. Let η denote the number of the block containing $v_{n_k-\mu}$, $\eta \le k-1$. Then we have for **NEXTXY**

$$V_k = \tilde{V}_k - \hat{V}_{k-1}M_{k-1}\tilde{V}_k - \hat{V}_{k-2}M_{k-2}\tilde{V}_k - \cdots - \hat{V}_\eta M_\eta \tilde{V}_k, \tag{6.1}$$

$$\begin{aligned}W_k = \tilde{W}_k - \hat{W}_{k-1}T_{k-1}S_{k-1}^{-1}M_{k-1}\tilde{V}_k - \hat{W}_{k-2}T_{k-2}S_{k-2}^{-1}M_{k-2}\tilde{V}_k\\ - \cdots - \hat{W}_\eta T_\eta S_\eta^{-1}M_\eta\tilde{V}_k,\end{aligned} \tag{6.2}$$

$$\begin{aligned}\hat{V}_k &= V_k S_k^{-1},\\ \hat{W}_k &= W_k T_k^{-1},\end{aligned}$$

$$\tilde{v}_{n+1} = s_n A\hat{v}_n(-s_n\zeta_{n-n_k}\hat{v}_n + s_{n-1}\eta_{n-n_k}\hat{v}_{n-1}), \tag{6.3}$$
$$\tilde{w}_{n+1} = t_n A^T\hat{w}_n(-t_n\zeta_{n-n_k}\hat{w}_n + t_{n-1}\eta_{n-n_k}\hat{w}_{n-1}), \tag{6.4}$$

$$\begin{aligned}v_{n+1} &= \tilde{v}_{n+1} - \hat{V}_{k-1}M_{k-1}\tilde{v}_{n+1} - \hat{V}_k M_k \tilde{v}_{n+1},\\ w_{n+1} &= \tilde{w}_{n+1} - \hat{W}_{k-1}T_{k-1}S_{k-1}^{-1}M_{k-1}\tilde{v}_{n+1} - \hat{W}_k T_k S_k^{-1} M_k \tilde{v}_{n+1},\end{aligned}$$

$$\begin{aligned}\hat{v}_{n+1} &= v_{n+1}/s_{n+1},\\ \hat{w}_{n+1} &= w_{n+1}/t_{n+1}.\end{aligned}$$

The terms in parentheses in (6.3) and (6.4) are not strictly necessary, since one then biorthogonalizes against these vectors, but they could enhance the numerical stability. If the size of the current block is at most the size of the previous block plus 1, then we have

$$\begin{aligned}n_{k+1} - n_k &\le n_k - n_{k-1} + 1 \Leftrightarrow n_{k-1} \le 2n_k - n_{k+1} + 1 \Leftrightarrow\\ n_{k-1} &\le n_k - (n_{k+1} - n_k - 1) \Leftrightarrow n_{k-1} \le n_k - \mu_{max} \Leftrightarrow\\ \eta &= k-1,\end{aligned}$$

which shows that under these conditions, the formulas (6.1) and (6.2) for V_k and W_k reduce to just two terms. Here μ_{max} is the largest value of μ. Finally, we note that the products of the form $M_j\tilde{V}_k$ that appear above have the structure

$$(\hat{W}_j^T\hat{V}_j)^{-1}\hat{W}_j^T\tilde{V}_k = (\hat{W}_j^T\hat{V}_j)^{-1}\begin{bmatrix}0 & \cdots & \cdots & 0\\ \vdots & & \iddots & *\\ \vdots & \iddots & \iddots & \vdots\\ 0 & * & \cdots & *\end{bmatrix},$$

since the regular vector is biorthogonal to all previous blocks.

Computing $\hat{H}^{(n_k-1)}$ is trickier, since we modify the vectors used in the inner recursion. Nevertheless, $\hat{H}^{(n_k-1)}$ retains the block tridiagonal structure (3.5). We will show this by induction. Assume that (3.3) and (3.5) hold for all $n = n_l - 1$, $l = 1, 2, \ldots, k-1$. Thus

$$A\hat{V}_l = \hat{V}_{l-1}\hat{\beta}_l + \hat{V}_l\hat{\alpha}_l + \hat{V}_{l+1}\hat{\gamma}_{l+1}, \quad l = 1, 2, \ldots, k-1. \tag{6.5}$$

We need to show that (6.5) also holds for $l = k$. For $\hat{V}_k$, we have

$$\begin{aligned}
A\hat{V}_k &= AV_kS_k^{-1} \\
&= A\left(\tilde{V}_k - \hat{V}_{k-1}M_{k-1}\tilde{V}_k - \hat{V}_{k-2}M_{k-2}\tilde{V}_k - \cdots - \hat{V}_\eta M_\eta \tilde{V}_k\right)S_k^{-1} \\
&= \left(\tilde{V}_k\tilde{\alpha}_k + [0 \quad \cdots \quad 0 \quad \tilde{v}_{n+1}]\right)S_k^{-1} - A\hat{V}_{k-1}M_{k-1}\tilde{V}_kS_k^{-1} - \cdots \\
&\quad - A\hat{V}_\eta M_\eta \tilde{V}_k S_k^{-1} \\
&= \left(\hat{V}_kS_k + \hat{V}_{k-1}M_{k-1}\tilde{V}_k + \cdots + \hat{V}_\eta M_\eta \tilde{V}_k\right)\tilde{\alpha}_kS_k^{-1} \\
&\quad + [0 \quad \cdots \quad 0 \quad \hat{v}_{n+1}s_{n+1} + \hat{V}_{k-1}M_{k-1}\tilde{v}_{n+1} + \hat{V}_kM_k\tilde{v}_{n+1}]\,S_k^{-1} \\
&\quad - \left(\hat{V}_{k-2}\hat{\beta}_{k-1} + \hat{V}_{k-1}\hat{\alpha}_{k-1} + \hat{V}_k\hat{\gamma}_k\right)M_{k-1}\tilde{V}_kS_k^{-1} \\
&\quad - \left(\hat{V}_{k-3}\hat{\beta}_{k-2} + \hat{V}_{k-2}\hat{\alpha}_{k-2} + \hat{V}_{k-1}\hat{\gamma}_{k-1}\right)M_{k-2}\tilde{V}_kS_k^{-1} \\
&\quad - \left(\hat{V}_{k-4}\hat{\beta}_{k-3} + \hat{V}_{k-3}\hat{\alpha}_{k-3} + \hat{V}_{k-2}\hat{\gamma}_{k-2}\right)M_{k-3}\tilde{V}_kS_k^{-1} \\
&\quad - \cdots - \left(\hat{V}_{\eta-1}\hat{\beta}_\eta + \hat{V}_\eta\hat{\alpha}_\eta + \hat{V}_{\eta+1}\hat{\gamma}_{\eta+1}\right)M_\eta\tilde{V}_kS_k^{-1} \\
&= -\hat{V}_{\eta-1}\left(\hat{\beta}_\eta M_\eta \tilde{V}_kS_k^{-1}\right) + \cdots \\
&\quad + \hat{V}_{k-2}\left(\tilde{\alpha}_kM_{k-2} - \hat{\beta}_{k-1}M_{k-1} - \hat{\alpha}_{k-2}M_{k-2} - \hat{\gamma}_{k-2}M_{k-3}\right)\tilde{V}_kS_k^{-1} \\
&\quad + \hat{V}_{k-1}\Big(M_{k-1}\tilde{V}_k\tilde{\alpha}_k + [0 \quad \cdots \quad 0 \quad M_{k-1}\tilde{v}_{n+1}] - \hat{\alpha}_{k-1}M_{k-1}\tilde{V}_k \\
&\qquad - \hat{\gamma}_{k-1}M_{k-2}\tilde{V}_k\Big)S_k^{-1} \\
&\quad + \hat{V}_k\left(S_k\tilde{\alpha}_k + [0 \quad \cdots \quad 0 \quad M_k\tilde{v}_{n+1}] - \hat{\gamma}_kM_{k-1}\tilde{V}_k\right)S_k^{-1} \\
&\quad + \hat{V}_{n+1}\left(S_{n+1}[0 \quad \cdots \quad 0 \quad e_1]\,S_k^{-1}\right),
\end{aligned}$$

where e_1 is the first column of the identity matrix. Suppose now we multiply the equation on the left by $\hat{W}_{\eta-1}$. Then on the right hand side, only the $\hat{V}_{\eta-1}$ term survives, as all the other blocks are biorthogonal to $\hat{W}_{\eta-1}$ by construction. In addition, on the left hand side, the term is also zero, again by biorthogonality. Hence, the coefficient of $\hat{V}_{\eta-1}$ in the above expression is zero. Similarly, by multiplying on the left by $\hat{W}_\eta$, $\hat{W}_{\eta+1}$, $\ldots$, $\hat{W}_{k-2}$, one shows that in fact all the coefficients up to $\hat{V}_{k-2}$ are zero. Hence, we are left with:

$$\begin{aligned} A\hat{V}_k &= \hat{V}_{k-1}\Big(M_{k-1}\tilde{V}_k\tilde{\alpha}_k + [0 \ \cdots \ 0 \ \ M_{k-1}\tilde{v}_{n+1}] - \hat{\alpha}_{k-1}M_{k-1}\tilde{V}_k \\ &\qquad - \hat{\gamma}_{k-1}M_{k-2}\tilde{V}_k\Big)\, S_k^{-1} \\ &\quad + \hat{V}_k\Big(S_k\tilde{\alpha}_k + [0 \ \cdots \ 0 \ \ M_k\tilde{v}_{n+1}] - \hat{\gamma}_k M_{k-1}\tilde{V}_k\Big)\, S_k^{-1} \\ &\quad + \hat{V}_{n+1}\left(S_{n+1}\,[0 \ \cdots \ 0 \ \ e_1]\, S_k^{-1}\right). \end{aligned}$$

The coefficient matrices $\hat{\alpha}_k$, $\hat{\beta}_k$, and $\hat{\gamma}_{k+1}$ are easily recognized. Note that the matrix $\hat{\alpha}_k$ which appears on the diagonal of $\hat{H}$ is a comrade matrix (from the $\tilde{\alpha}_k$ and $\tilde{v}_{n+1}$ terms), but in addition, the first row fills in (from the $\hat{\gamma}_k$ term). This is different from the sequential case, where the diagonal blocks are just comrade matrices.

Finally, we would like to stress that — even if long recurrences (of more than two terms) occur in the updates (6.1) and (6.2) for V_k and W_k — the matrices AV_k, V_{k+1}, V_k, and V_{k-1} (and similarly for the W matrices) are still connected via a three-term recursion, cf. (6.5). Both the sequential and the block algorithm generate upper Hessenberg matrices $H^{(n)}$ with the same block tridiagonal structure. This is important if the block algorithm is used for eigenvalue computations or in conjunction with the QMR approach (see Section 9) for solving linear systems.

7. CONCLUDING REMARKS

We have presented the details of an implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. Our implementation can handle look-ahead steps of any length and is not restricted to steps of length 2, as earlier implementations are. Also, the proposed algorithm requires roughly the same number of inner products as the standard Lanczos process without look-ahead. It was our intention to develop a robust algorithm which can be used in a black box solver.

This paper is continued in Part II [6]. There, a robust black box solver for non-Hermitian linear system, the QMR algorithm based on the look-ahead Lanczos algorithm, is presented. Also, in [6], numerical experiments, both with the implementation of the look-ahead Lanczos method proposed here and with the QMR algorithm, are reported. Finally, for the case of real nonsymmetric matrices A, the FORTRAN programs for these algorithms are listed in Part II.

In the future, we plan to also provide FORTRAN codes for complex non-Hermitian matrices. Often when complex matrices arise in practical applications, they are complex symmetric. For this important special case, the look-ahead Lanczos algorithm can be arranged such that left and right Lanczos vectors coincide, and thus work and storage is halved. We also plan to provide FORTRAN codes for the resulting complex symmetric variant of the look-ahead Lanczos algorithm.

In the present paper, we have only outlined a block version of the look-ahead Lanczos algorithm which appears to be better suited for parallel computers. Details of an actual implementation and experiments with it will be presented elsewhere.

REFERENCES

[1] Boley, D., Elhay, S., Golub, G. H., and Gutknecht, M. H. Nonsymmetric Lanczos and finding orthogonal polynomials associated with indefinite weights. Numerical Analysis Report NA–90–09, Stanford, August 1990.

[2] Boley, D., and Golub, G. H. The nonsymmetric Lanczos algorithm and controllability. Numerical Analysis Report NA–90–06, Stanford, May 1990.

[3] Brezinski, C. *Padé-Type Approximation and General Orthogonal Polynomials.* Birkhäuser, Basel, 1980.

[4] Draux, A. *Polynômes Orthogonaux Formels – Applications.* Lecture Notes in Mathematics, vol. 974, Springer, Berlin, 1983.

[5] Freund, R. W. Conjugate gradient type methods for linear systems with complex symmetric coefficient matrices. Technical Report 89.54, RIACS, NASA Ames Research Center, December 1989.

[6] Freund, R. W., and Nachtigal, N. M. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, Part II. Technical Report 90.46, RIACS, NASA Ames Research Center, November 1990.

[7] Freund, R. W., and Nachtigal, N. M. QMR: a quasi-minimal residual method for non-Hermitian linear systems. Technical Report, RIACS, NASA Ames Research Center, 1990, in preparation.

[8] Gragg, W. B. Matrix interpretations and applications of the continued fraction algorithm. *Rocky Mountain J. Math. 4* (1974), 213–225.

[9] Gragg, W. B., and Lindquist, A. On the partial realization problem. *Linear Algebra Appl. 50* (1983), 277–319.

[10] Gutknecht, M. H. A completed theory of the unsymmetric Lanczos process and related algorithms, Part I. IPS Research Report No. 90–10, Zürich, June 1990.

[11] Gutknecht, M. H. A completed theory of the unsymmetric Lanczos process and related algorithms, Part II. IPS Research Report No. 90–16, Zürich, September 1990.

[12] Joubert, W. Lanczos methods for the solution of nonsymmetric systems of linear equations. In *Proceedings of the Copper Mountain Conference on Iterative Methods, April 1–5, 1990.*

[13] Kung, S. Multivariable and multidimensional systems: analysis and design. Ph.D. Dissertation, Stanford University, Stanford, June 1977.

[14] Lanczos, C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand. 45* (1950), 255–282.

[15] LANCZOS, C. Solution of systems of linear equations by minimized iterations. *J. Res. Natl. Bur. Stand. 49* (1952), 33–53.

[16] MANTEUFFEL, T. A. The Tchebychev iteration for nonsymmetric linear systems. *Numer. Math. 28* (1977), 307–327.

[17] PARLETT, B. N. Reduction to tridiagonal form and minimal realizations. Preprint, Berkeley, January 1990.

[18] PARLETT, B. N., TAYLOR, D. R., AND LIU, Z. A. A look-ahead Lanczos algorithm for unsymmetric matrices. *Math. Comp. 44* (1985), 105–124.

[19] TAYLOR, D. R. Analysis of the look ahead Lanczos algorithm. Ph.D. Dissertation, University of California, Berkeley, November 1982.

[20] WILKINSON, J. H. *The Algebraic Eigenvalue Problem.* Oxford University Press, Oxford, 1965.